REMARKS

Reconsideration is respectfully requested.

The Specification has been objected to containing a typographical error on page 21, line 26. As suggested by the Examiner, "shareing" has been corrected to --sharing--, and withdrawal of this objection is therefore respectfully requested.

Claims 10-18 are pending in the present application before this amendment. By the present amendment, Claims 10 has been amended. No new matter has been added.

Claims 10 and 11 stand rejected under 35 U.S.C. § 103(a) as being obvious over a reference "Fault-Tolerance for Communicating Multdatabase Transactions (Kuhn94) in view of U.S. Patent No. 5,743,898 (He).

In response to these rejections, Claim 10 has been amended.

The support for the first limitation --so that the network of computers operates like a single global computer where the addition of new processes and sites is dynamic-- is found in the Specification, page 8, the last paragraph and other paragraphs cited below:

> "By all coordination servers behaving as a worldwide, uniform operating system – where all coordination servers are identical, the advantage gained inter alia that a unique treatment is guaranteed, that the number and location of the computers – or agents – is irrelevant (the system provides a specific network), and that in case of lost objects at least those parts which are independent of lost objects can be repaired or be saved as a consistent unit."

From this disclosure, it follows that the number of computers may change. Even if one computer is lost the global operating system can continue to work saving all "healthy" parts, i.e. maintaining the operation at least for the "healthy" objects.

> "Thus, the invention provides a system or network, operating in a very specific way wher , the distributed computers operate according to a common, global

operating system in summary, the invention makes the coordination of software systems, services and resources considerably easier." (Specification page 12, lines 13 to 18)

This addresses the notion of a single "super" computer. All computers work together like a single global operating system that behaves like a single, global "super" computer. This obviously goes beyond a "global shared space" abstraction.

"For the programmer this system looks like a globally available space, even though, in reality, it is distributed over a number of sites, comparable to a large distributed database." (Specification page 20, lines 16-20)

"Due to this global operating system 24, it does not matter for the user whether a process runs locally or at a remote place;" (Specification page 25, lines 2-4)

"The addition of new producers and consumers (at distributed sites) is dynamic ("dynamic scale-up")" (Specification page 81, lines 16-17)

The above passages particularly stress that in all of the presently claimed system, the abstraction of the space always remains the same. It gives the programmer the impression to work with one single "super" computer only – in the described specific way. The programmer uses the same functions to access local and remote resources. The dynamic change of the amount of processes (=services) and thus of distributed sites (=computers that build up the super computer) is possible.

Accordingly, the difference between the notion of a "virtually or globally shared space" and the feature of a "global operating system" is made more clear in Claim 10 by the cited amendment: "...so that the distributed computers operate like a single global computer where the addition of new processes (services) and sites is dynamic".

The "globally shared space" refers to the abstraction offered to programmers of the distributed resources in the network. Contrary to this, according to the present application, the "global operating system" denotes the realization in form of a so-called "super" computer. This could be on one single computer or on many computers in the network. The amount of

computers can dynamically be changed without violating the idea of the "single gl bal super computer." Such a super computer resembles the idea of GRID computing, which, however, traditionally employs a client/server oriented approach (offering the abstraction of a "global service architecture"), instead of a globally shared memory absuraction.

The present global operating system uses the described distribution strategies to manage the abstraction of the shared space. Other abstractions could be possible – this again implies that the notion of a global operating system is not identical with the notion of a global shared space.

The support for the second limitation of Claim 1 --where different consistency models are supported for an updateable object-- is shown by the following paragraphs of the Specification page 50, bottom to page 51, top; and/or page 30, bottom to page 31, etc.

This second limitation has been added to Claim 1 in view of the Examiner's specific objections under the Office Action, paragraph 5, and this amendment should make the distinguishable concept of an updateable object and its logical "time stamp" more clear.

The added limitation --where different consistency models are supported for an updateable object-- makes clear that how an object that is accessed at the presently claimed system can be controlled by the programmer. This is different from the distribution strategy used for the object. At the presently claimed system, several programmers can employ different consistency policies on the same object. This is primarily done by making use of the logical read time stamp–which obviously has a completely different meaning in the coordination model when compared with that the time stamp described by He. He's system will always return the most recent value of the object, whereas in the present coordination model, the logical time stamp of the updateable object serves to control the consistency model used for reading this object. The read function uses a logical read time stamp that says

that the next value of the object with a logical time stamp larger than the given logical read time stamp suffices the request.

Further, the consistency control can also be done by using the try commit function in addition to the logical read time stamp or by using the non-transactional read.

The actual network behavior depends on the distribution strategy used.

In connection therewith, the following examples are submitted to make the concept of the presently claimed invention even more clear:

(1) The weakest form would be to use a "lazy" distribution model and to specify read time stamp = 0 and to use the blocking non-transactional read. In this case the object copy from the local cache is taken, if already there.

(2) The strongest form is to use the logical read time stamp returned by the last read call for reading, and to use the blocking transactional read. This assures that a new value is read. At commit time the verification is done that the read value is still the most recent one.

(3) Another example would be to use a blocking transactional read with a certain read time stamp, saying that the next value after the given read time stamp would be fine and using the try commit function. If the read value is still valid, the commit will finish, otherwise it will report that the read failed and now the programmer may decide, whether the read value suffices or not. If yes, then the programmer may simply cancel the read request from the transaction and proceed with commit. If no, the programmer may read again the updateable object with a larger logical read time stamp,

(4) Many further consistency model scenarios are possible.

Claims 13-18 stand rejected under 35 U.S.C. § 103(a) as being obvious over Kuhn94 and He, and in view of a reference "Logic Based and Imperative Coordination Languages" (Frost).

In response, it is respectfully submitted that Claims 13-18 are considered to be allowable at least since they depend on Claim 10, as amended, that is now considered to be in condition for allowance.

For the reasons set forth above, Applicant respectfully submits that Claims 10-18, pending in this application, are in condition for allowance over the cited references. This amendment is considered to be responsive to all points raised in the Office Action. Accordingly, Applicant respectfully requests reconsideration and withdrawal of the outstanding rejections and earnestly solicits an indication of allowable subject matter. Should the Examiner have any remaining questions or concerns, the Examiner is encouraged to contact the undersigned attorney by telephone to expeditiously resolve such concerns.

Respectfully submitted,

Dated: _____

Richard J. Streit, Reg. No. 25,765
c/o Ladas & Parry
224 South Michigan Avenue
Chicago, Illinois 60604
(312) 427-1300

7/21/03

DOCKET: CU-1867 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application:	Eva Kuhn	
Serial No:	9/269,485	GRP ART UNIT: 2126
Filed:	March 29, 1999	Ex.: Li B. Zhen
For:	COORDINATION SYSTEM	AMENDMENT AFTER FINAL REJECTION

SPECIFICATION - MARKED-UP VERSION
(IN THE REVISED FORMAT)

Please amend the paragraph appearing on page 21, line 26 to page 22, line 2 as set forth below:

In order to give to all processes ~~shareing~~ sharing communication objects to a consistent view, values can be written only within transactions into the globally shared space.

DOCKET: CU-1867 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application:	Eva Kuhn	
Serial No:	9/269,485	GRP ART UNIT: 2126
Filed:	March 29, 1999	Ex.: Li B. Zhen
For:	COORDINATION SYSTEM	AMENDMENT AFTER FINAL REJECTION

CLAIMS - MARKED-UP VERSION
(IN THE REVISED FORMAT)

Please amend Claim 10 as set forth below:

1-9. (cancelled)

10. (previously added and currently amended) A system for the coordination of distributed programs, services and data by using application programs in a network of computers where coordination servers (CoKe) are running which serve local software systems (LSYS), where shared objects are used as communication objects to exchange messages and transactions are used to realized communication, said communication objects being uniquely identified by object identification numbers (OID), and only processes processing a reference to a communication object are granted access to it via the corresponding local coordination server, with the local software systems being at least extended by functions for the control of transactions, for the creation, reading and writing of communication objects, and for the creation and supervision of uniquely identified processes, and with the communication objects being administrated by means of replication strategies, wherein

all coordination servers are identical regarding their basic functionality for distributed object, transaction and process management, and taken together, form a

global operating system, so that the network of computers operates like a single global super computer where the addition of new processes and sites is dynamic,

at least some of the objects are updateable objects, the functions provided for the extension of the local software systems provide a transactional blocking read of the updateable object and the processes are granted access to passed communication objects where different consistency models are supported for an updateable object, and

distribution strategies are provided for the administration of communication objects, with the application programs not depending on said distribution strategies, and which distribution strategies are selectable at least with respect to the recoverability or non-recoverability of communication objects and processes.

11. (previously added) A system according to Claim 10, wherein when choosing the respective distribution strategy, a basic strategy is selected in combination with additional, optional strategy flags.

12. (previously added) A system according to Claim 11, wherein the local software systems can be started by the corresponding coordination server.

13. (previously added) A system according to claim 12, wherein communication objects, to which no locally running process possesses a reference any more, are automatically cleared by the corresponding coordination server or can be explicitly freed.

14. (previously added and amended) A system according to claim 13, wherein heterogeneous transactions or subtransactions are distributed to different sites (X,Y,Z) via the coordination servers which, taken together, behave as a global operating system.

15. (previously added) A system according to claim 14, wherein a non-blocking transactional read is provided for updateable objects.

C1

16. (previously added) A system according to claim 14, wherein the writing into an object, the starting of a subtransaction, the distribution of part of a transaction to another site, the specification of a compensation action or of an on-commitment action are provided as transactional predicate.

17. (previously added) A system according to claim 16, wherein an on-commitment action is started as a computation if it is sure that a transaction will commit.

18. (previously added) A system according to claim 17, wherein among the functions for transactions a programmable backtracking of transactional operations, e.g. reading or writing of communication objects, is provided to be able to dynamically repair faults or failures in the transactions.